I wanted to clear some of the confusion surrounding recent posts and articles discussing changes back to the VB.NET language. We've spent the past month or so reviewing the changes we made to Visual Basic.NET to ensure that we are shipping a version of VB that would provide power and cross-language interop as well as familiarity with VB6. We reviewed change requests from Visual Basic programmers around the world and while we reaffirmed many of the changes that we decided upon more than a year and a half ago, there were a handful that we found would need to be changed back to VB's original behavior in this release. The goal of these changes is to minimize *semantic* inconsistencies between VB6 and Visual Basic.NET. This is particularly crucial in going from VB6 to VB.NET, since many developers will be working with both VB6 and VB.NET code bases. The ability for these developers to look at a single line of VB code and not think about whether that line represents VB6 vs. VB.NET semantics is a key element to making the knowledge upgrade to VB.NET.

The changes to the language include:

1. The value of True: When Visual Basic developers coerce “True” into a numeric value, the value they receive will be -1 (as VB has always done) , not 1This applies to both explicit conversions, using CInt( ), CDec( ), etc., as well as implicit conversions:

```
Dim b As Boolean = True
Dim i As Integer = b                ‘ i = -1
```

It is important to note, that when a boolean in VB is passed to the .NET Framework or to another .NET language, it will be treated as 1 to maintain interoperability between languages. This change offers the best of both worlds - VB developers get expected behavior when working within VB and developers working with multiple .NET languages get seamless behavior across languages.

2. Behavior of And/Or/Not/XOr - The functionality of these operators will be returned to bitwise operations instead of purely logical (meaning they will work as they always have in VB). As a result, there is no need to keep the BitAnd, BitOr, and BitNot operators that were planned for VB.NET. In addition, the operator precedence will be restored to these operators as it was in Visual Basic 6. Finally, these operators will not short-circuit. Instead, to enable developers to show explicit intent in their code, we will introduce two new operators, “AndAlso” and “OrElse” that will provide short-circuiting behavior. Again, these changes provide consistency with VB6 without sacrificing functionality in VB.NET.

3. The declaration of Arrays - When arrays are declared in Visual Basic.NET, they will be declared by specifying the upper bound, not the size (as has always been the case in VB).

```
Dim a(5) As Integer       ' produces a 6 element array, 0 through 5
```

Do any of these changes break language interoperability with other .NET languages or the Framework? No.
Will these changes delay the ship of Visual Studio.NET? No. The changes are currently being implemented and will be part of Beta 2 (due out this summer).
Are these *major* changes to the language? No. Will they keep VB.NET developers from taking advantage of the .NET Framework, Web Services, ASP.NET, cross-language inheritance and debugging, etc, etc, etc? No, but they do provide benefits to the existing VB developer base who are either upgrading existing code to VB.NET, or who will be maintaining both VB6 and VB.NET code bases.

In response to custom feedback, the VB team is making the following language changes in order to ease our users migrating from VB6 to VB.NET.

Changes to Boolean/Logical Operators

- The BitAnd, BitOr, and BitNot operators are being removed.
- The And, Or, Xor, and Not operators of VB6 will be restored to VB.NET. These operators work on both numeric and Boolean types. When used on numeric types they perform a bitwise operation and when used on Boolean types they perform a Boolean operation. The precedence of these operators is the same as the existing And, Or, Xor, and Not operators in VB.NET. None of these operators short-circuit.
- When the Boolean value True is converted to integer it will map to -1 rather than 1
- The short-circuiting Boolean And operator will be supported with the name AndAlso
- The short-circuiting Boolean Or operator will be supported with the name OrElse

Also the meaning of the number specified when creating a new array instance will change from being the size of the array dimension (which is the meaning used in C) to being the upper bound of the array dimension (which is the meaning used in VB6). So, for example, "New Integer(10)" will create an array from 0 to 10 rather than an array from 0 to 9. Similarly "Dim x(10) As Integer" would create an array from 0 to 10.

An important note about the short-circuiting AndAlso and OrElse operators. DO NOT use search and replace in sample code and snippets to replace the And operator with AndAlso and the Or operator with OrElse. These operators are unfamiliar to VB users and their use in samples will make it more difficult to for them to understand the example. You should only use these operators when absolutely necessary and when you use them you should add a comment indicating their intent.

In addition to these breaking changes VB will be removing support for the attribute syntax supported in Beta1. For details on this change see the attached mail.

Also attached is the schedule for the changes and coordination of these changes across the division.

For those interested there is a tool available to automatically update code to change the value used when creating arrays. For example it will change “New Integer(10)” to “New Integer(10 – 1)”. This is only

useful for updating test code since the result is ugly. Please contact Cmccoll for details.